Parameterized Algorithms WS 2021
Prof. Dr. P. Rossmanith
Dr. E. Burjons, M. Gehnen, H. Lotze, D. Mock

RWTH AACHEN UNIVERSITY

Date: November 14th, 2021

Exercise Sheet with solutions 04

Task T12

The problem 3-Colorability is defined as follows: Given a graph G decide if it is possible to assign every node of G one of three colors, such that no two nodes with the same color are adjacent. This problem is fpt parameterized by the treewidth of the graph. Give an algorithm that solves the problem given a tree decomposition of width w in time $O(3^w \cdot w \cdot n)$.

Solution

First convert the tree decomposition into a nice tree decomposition $\mathcal{T}$ in linear time. If X is a bag of the tree decomposition, then let V_X be all nodes which are contained in X and in bag which is a descendant of X in T. We will solve this in the usual way, by computing a table for every bag which will contain entries as follows: For all possible 3-colorings c of $G[V_X]$ we want an entry $c(X)$, i.e. the colors of c for the nodes in X, in the table for the bag X. Such a table can have at most 3^w entries. Having such a table for the root would provide the solution for the decision problem, since by the definition of the table the graph is 3-colorable iff the table for the root bag is not empty. Assume we want to generate the table for some bag X. Let us do it case by case depending on what X can be:

leaf bag Generate a table that contains three entries, one for every color we can give to the only node in the bag.

forget bag Delete the forgotten node from every entry and delete duplicates.

introduce bag When a node x is introduced go through every coloring c of the nodes of the bag in the table of the child of X. If x does not have a neighbor in the bag with color c' add new a entry extending c with x colored with the color c'. This is correct since by the properties of tree decompositions x can only have neighbors which are contained in X in the graph $G[V_X]$.

join bag Only keep the entries which are also contained in both tables of the child bags of X.

Since these operation suffice to compute the table for the root bag in a bottom up fashion, all operations take time $O(3^w)$ for tables which at most 3^w entries and a nice tree decomposition has at most $w \cdot n$ bags it follows that the running time is $O(3^w \cdot w \cdot n)$.

Task T13

Design a fixed parameter algorithm for finding an $k \times k$-grid subgraph in a graph that is taken from a graph class with maximal degree d. The parameter is k. Analyse the running time.

Solution

For every vertex we calculate the $2k$ neighborhood. Fining one neighborhood takes at most time d^{2k}. which is also a bound on its size. We then check if this neighborhood is a $k \times k$ grid in time $d^{2k}\binom{d^{2k}}{k}k!$ (checking for every subset if it contains a grid). We repeat this for every vertex which adds a factor of n.

Task T14
Give a parameterized algorithm that decides if a graph G contains k many vertex disjoint claws as induced subgraph. A claw is a $K_{1,3}$.

Solution
We solve it by color coding: Give every vertex one of k colors independently uniformly at random. Let G_i be the subgraph of G induced on all vertices with color i. We can find a claw in G_i in $O(n^4)$ by trying all subsets of size 4. If every G_i contains a claw we answer yes, otherwise no.
The probability that k vertex disjoint claws are colored in a way such that the algorithm answers yes, that is one color per claw and all claws different colors, is $p = k!/k^{4k}$. The probability that the algorithm answers wrong $1/p$ times is $(1-p)^{1/p} \leq 1/e$, because $(1+c/n)^n \leq e^c$ for all c. This means we have a constant failure probability after $O(k^{4k}/k!) = O^*(e^{4k})$ steps.

Task T15
Design a fixed parameter algorithm for finding a cycle of length *at least* k in an arbitrary graph G.

Solution
If the treewidth of G is at most k we can solve this problem via Courcelle's Theorem. If the treewidth of G is larger, then it has to contain a large grid as a minor and is automatically a yes-instance.

Task H8 (5 credits)
The problem Dominating Set is defined as follows: Given a graph G find the smallest set $S \subseteq V(G)$ such that every node of G is either in S or has a neighbor in S. This problem is fpt parameterized by the treewidth of the graph. Give an algorithm that solves the problem given a tree decomposition of width w in time $O(9^w \cdot w \cdot n)$.

Solution
We will use the same notation as in for the solution of T1. We will also solve it in a very similar manner. The interpretation of the tables for the bags will this time be somewhat more complicated: In the table we will have an entry which represent partitions of the current bag X intro three sets S, D and F and are associated with a number s. Such an entry is in the table only if there exists a partial dominating set of $G[V_X]$ of size s where every node not in the bag X is dominated, the nodes in S are in the dominating set, the nodes in D are dominated but not in the dominating set and the nodes in F are not dominated. Because we are trying to minimize the size of the dominating set between two entries with the same sets but different sizes s we will only keep the entry where the size is smaller. This means that, since there are only 3^w ways to partition a set of size w into three sets, the tables have at most 3^w entries. If we can compute such a table we would be able to read the size if a minimal dominating set by taking the minimum size over all the entries where F is empty. We can then again describe the operations needed to compute these tables in a bottom up fashion a nice tree decomposition.

leaf bag Create three entries, such that x is contained only in S, D and F respectively. If the nodes is in S set s to one, otherwise to zero.

forget bag Take the whole table of the child bag. Remove all the entries where the forgotten node x is contained in F. Then remove x from every set in the remaining table. We can do this because we only want entries that represent partial dominating sets that dominate every node which is not in the bag X.

introduce bag Take every entry (S, D, F, s) and create three new entries as follows:

- Put x in S, all the neighbors of x in X in the set D and increase s by one. This is correct since by the properties of tree decompositions x can not dominate anything in $G[V_X]$ which is not in the bag X.
- Put x in D if x has a neighbor in S. Keep s.
- Put x in F if x does not have a neighbor in S. Keep s.

join bag Take the tables t_1 and t_2 from the children bags of X. For every pair of entries $(S_1, D_1, F_1, s_1) \in t_1$ and $(S_2, D_2, F_2, s_2) \in t_2$ generate a new entry (S, D, F, s) where

- $S = S_1 \cup S_2$,
- $D = (D_1 \cup D_2) \setminus (S_1 \cup S_2)$,
- $F = X \setminus (S \cup D)$ and
- $s = s_1 + s_2 - |D_1 \cap D_2|$.

This is correct since by the properties of tree decompositions when merging two entries we only have to take care that the solution agrees on the join bag. Since we have to look at all pairs this step takes time $O((3^w)^2) = O(9^w)$.

By using this algorithm on a nice tree decomposition, since the join operation has the worst running time, we have an algorithm with the desired running time.

Task H9 (5 credits)
The Triangle Packing problem is defined as follows: given a graph $G = (V, E)$ and an integer k, decide whether G has k vertex-disjoint 3-cycles. Use the idea of randomly coloring the vertices of G with k colors to enable easy detection of vertex-disjoint triangles. What is the running time of your algorithm? And what is the failure probability?

Solution
Assume that the graph G has a set of k vertex-disjoint triangles. Call a random coloring $c\colon V(G) \to \{1, \ldots, k\}$ of the vertices *proper* if all vertices of a triangle receive the same color but different triangles receive different colors. Given a proper coloring of a yes-instance of the problem, one can find the triangles but searching for them in the graphs induced by the vertices of each color class, in turn. The probability that a coloring is proper is at least

$$\frac{k^{n-3k} \cdot k!}{k^n} = \frac{k!}{k^{3k}} > \frac{1}{k^{3k}} \cdot \left(\frac{k}{e}\right)^k = \left(\frac{1}{k^2 e}\right)^k.$$

The randomized algorithm repeats the following steps in an infinite loop: it randomly colors the vertices of the graph using colors $\{1, \ldots, k\}$; it then checks Since one obtains a proper coloring with probability $(k^2 e)^{-k}$, the expected running time is $O((k^2 e)^k \cdot \text{poly}(n))$.

Task H10 (10 credits)
Design a fixed parameter algorithm for finding an $k \times k$-grid as a *minor* in a graph. The parameter is k.
Hint: Design first an algorithm which says whether there exists a $k \times k$-grid as minor.

Solution
DRAFT
There is a function f such that for every graph with treewidth $f(k)$ it contains a $k \times k$ grid as a minor.